



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/839,624	04/19/2001	Shubhendu S. Mukherjee	1662-37500 JMH (P00-3162)	2433
23505	7590	03/09/2004	EXAMINER	
CONLEY ROSE, P.C. P. O. BOX 3267 HOUSTON, TX 77253-3267			GERSTL, SHANE F	
			ART UNIT	PAPER NUMBER
			2183	

DATE MAILED: 03/09/2004

6

Please find below and/or attached an Office communication concerning this application or proceeding.

# Office Action Summary

Application No.

09/839,624

Applicant(s)

MUKHERJEE, SHUBHENDU S.

Examiner

Shane F Gerstl

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the corresponding address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 28 October 2002 and 22 April 2002.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-21 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-21 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 19 April 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☒ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date 2 and 3.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

### **DETAILED ACTION**

1. Claims 1-21 have been examined.

#### ***Papers Received***

2. Receipt is acknowledged of both information disclosure papers and the power of attorney paper submitted, where the papers have been placed of record in the file.

#### ***Specification***

3. The attempt to incorporate subject matter into this application by reference to the copending patent applications given on pages 1 and 2 of the specification is improper because the application numbers and filing dates have not been given. Appropriate correction including these application numbers and filing dates are required.

#### ***Claim Objections***

4. Claim 7 is objected to because of the following informalities: the dependent claim is separated by other dependent claims that do not depend on the same parent claim 4. The examiner will not change the order of the claims. See MPEP § 608.01(n).

Appropriate correction is required.

#### ***Claim Rejections - 35 USC § 112***

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 1-7 and 13 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

7. Claim 1 recites the limitation "the corresponding data load command" in line 10. There is insufficient antecedent basis for this limitation in the claim. A corresponding data load command has not been defined to this point and it is not inherent as existing in the trailing thread since though the trailing thread is redundant, the claim does not say what about the thread is redundant or specifically that the instructions in the thread are redundant. The examiner is taking the limitation to read, "a corresponding data load command."

8. Claim 13 recites the limitation "the fetch command" in line 6. There is insufficient antecedent basis for this limitation in the claim. A fetch command has not yet been defined. The examiner is taking the limitation to mean "a fetch command was executed".

***Claim Rejections - 35 USC § 102***

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

10. Claims 1-7, 15-17 and 19-21 are rejected under 35 U.S.C. 102(a) as being anticipated by Rotenberg (AR-SMT).

11. In regard to claim 1, Rotenberg discloses a computer system, comprising:

a. a pipelined, simultaneous and redundantly threaded ("SRT") processor comprising at least one data cache; In the second column of section 1.1, the system is shown to run two explicit copies of the program run concurrently on the

same processor resources. Further down it is shown that this is possible using simultaneous multithreading. Thus two copies of the same program (the term redundant is used on the top of the next page) are run in these simultaneous threads. Figure 5 shows that the system is broken up into pipeline stages and is thus pipelined. This figure, as well as figure 4, shows that the system includes a data cache or d-cache.

b. an I/O controller coupled to said processor, which in turn is coupled to at least one I/O device; Section 2.1.2 discusses a memory disambiguation unit of the processor (shown in figure 5) that controls memory. Memory is an I/O device since data is input from and output to it from the processor and thus there exists an I/O controller coupled to the processor and an I/O device.

c. a main system memory coupled to said processor; As shown above there is a system memory coupled to the processor through the disambiguation unit.

d. and wherein said SRT processor processes a set of instructions in a leading thread and also in a redundant trailing thread to detect transient faults in the computer system; Section 1.2 shows that the disclosed system comprises multiple threads (streams) that run the same program at the same time with one thread having a slight lag and thus a set of instructions is run in a leading and trailing thread.

e. and wherein when a data load command appears in the leading thread, the processor loads the requested data and replicates the value for the corresponding data load command in the trailing thread. The paragraph

immediately under figure 3 shows that a delay buffer contains data and control information from the first run of the program (on the leading thread or A-stream). This is a clear differentiation of the result data and other control information such as that for predictions. In the paragraph directly above figure 3, it is shown that source operands (data) of instructions are predicted. In the paragraph directly below the figure again, it is stated that the delay buffer provides the R-stream (trailing thread) with perfect data flow prediction. Therefore, perfect data operand prediction must be provided, and thus the data operands themselves provided (replicated as the control information) in this delay buffer for the trailing thread.

12. In regard to claim 2, Rotenberg discloses the computer system of claim 1 further comprising a load value queue; wherein when the processor loads the requested data, the processor stores the same data in the load value queue. As shown above, the loaded operand data is stored in a delay buffer. The first paragraph of section 1.2 shows that this buffer is in fact a queue and thus another appropriate name for the structure is a load value queue.

13. In regard to claim 3, Rotenberg discloses the computer system of claim 2 wherein the processor does not store the data value in the load value queue until the data load command in the leading thread commits. Figure 2 and the first paragraph of section 1.2 shows that the data stored in the load value queue (delay buffer) is pushed there when the results of the A-stream (leading thread) are committed.

14. In regard to claim 4, Rotenberg discloses the computer system of claim 2 wherein the load value queue is a FIFO buffer and wherein all data load commands in

the trailing thread are executed by the processor in their original, program order. The first paragraph of section 1.2 show that the load value queue is a FIFO delay buffer. Though section 1.3.1 details that instructions in the trailing thread (R-stream) may be executed highly in parallel due to perfect prediction, the instructions are not executed out-of-order, as can be seen in figure 3. Therefore the trailing thread executes instructions including the data load commands in program order.

15. In regard to claim 5, Rotenberg discloses the computer system of claim 2 wherein the data load command is a request for data in the data cache. As shown above, a data cache exists in the system. The load commands have been shown above to be requests for data operands. It is inherent that a data cache first requests data from a data cache because its whole purpose of existence is to be a fast location for data retrieval. Further proof is given in the included IEEE dictionary term for "hit" (shown to be a synonym for cache hit). Here it shows that referencing data in cache eliminates the need to go to a secondary storage.

16. In regard to claim 6, Rotenberg discloses the computer system of claim 2 wherein the data load command is a request for data in the main system memory. As shown above, a data cache exists in the system. The load commands have been shown above to be requests for data operands. It is inherent that a data cache first requests data from a data cache because its whole purpose of existence is to be a fast location for data retrieval. It is also inherent that if the data cache misses (the data is not in the cache), the data will be requested from the main system memory. Further proof is given in the included IEEE dictionary term for "hit" (shown to be a synonym for

cache hit). Here it shows that referencing data in cache eliminates the need to go to a secondary storage, thus meaning that on a miss, the secondary storage is referenced. Since the disclosure of Rotenberg only discloses two data storage locations, the data cache and main memory, both shown above, the secondary storage is main system memory and thus is referenced.

17. In regard to claim 7, Rotenberg discloses the computer system of claim 4 wherein if the load value queue becomes full, execution of instructions in the leading thread is temporarily halted to prevent more data values from entering the load value queue; and wherein if the load value queue becomes empty, execution of instructions in the second thread is temporary halted to allow more data values to enter the load value queue. Near the end of the summary section (section 4), it is disclosed that if the delay buffer (load value queue) is full, the R-stream (trailing thread) is given priority to access the fetch/dispatch pipeline, meaning that the A-stream or leading thread is temporarily halted in favor of the trailing thread so that it can issue and dispatch instructions. It is also disclosed that if the delay buffer (load value queue) is not full (where empty is an instance of this), the A-stream (leading thread) is given priority retire a trace, meaning that the R-stream or trailing thread is temporarily halted in favor of the leading thread when it can retire a trace of instructions.

18. In regard to claim 15, Rotenberg discloses a method of replicating data values in an SRT processor which can fetch and execute a program set in two separate threads so that each thread includes substantially the same instructions as the other thread, one of said threads being a leading thread and the other of said threads being a trailing



thread; In the second column of section 1.1, the system is shown to run two explicit copies of the program run concurrently on the same processor resources. Further down it is shown that this is possible using simultaneous multithreading. Thus two copies of the same program (the term redundant is used on the top of the next page) are run in these simultaneous threads. Section 1.2 shows that the disclosed system comprises multiple threads (streams) that run the same program at the same time with one thread having a slight lag and thus a set of instructions is run in a leading and trailing thread. the method comprising:

- a. accessing a data source to load a data value when the leading thread requests said data; It is inherent that when the leading thread requests data that it requests it from a data source of some sort.
- b. storing the data value in a load value queue; accessing the load value queue for the data value for corresponding data requests in the trailing thread.

The paragraph immediately under figure 3 shows that a delay buffer contains data and control information from the first run of the program (on the leading thread or A-stream). This is a clear differentiation of the result data and other control information such as that for predictions. In the paragraph directly above figure 3, it is shown that source operands (data) of instructions are predicted. In the paragraph directly below the figure again, it is stated that the delay buffer provides the R-stream (trailing thread) with perfect data flow prediction.

Therefore, perfect data operand prediction must be provided, and thus the data operands themselves provided (replicated as the control information) in this delay

buffer for the trailing thread. The first paragraph of section 1.2 shows that this buffer is in fact a queue and thus another appropriate name for the structure is a load value queue.

19. In regard to claim 16, Rotenberg discloses the method of claim 15 further comprising: executing the data requests in the trailing thread in program order. Though section 1.3.1 details that instructions in the trailing thread (R-stream) may be executed highly in parallel due to perfect prediction, the instructions are not executed out-of-order, as can be seen in figure 3. Therefore the trailing thread executes instructions including the data load commands in program order.

20. In regard to claim 17, Rotenberg discloses the method of claim 16 further comprising: storing the data values in the load value queue after the data requests in the leading thread execute. Figure 2 and the first paragraph of section 1.2 shows that the data stored in the load value queue (delay buffer) is pushed there when the results of the A-stream (leading thread) are committed (and thus have executed).

21. In regard to claim 19, Rotenberg discloses the method of claim 18 wherein the data source is a data cache. As shown in figures 4 and 5, a data cache exists in the system. The load commands have been shown above to be requests for data operands. It is inherent that a data cache first requests data from a data cache because its whole purpose of existence is to be a fast location for data retrieval. Further proof is given in the included IEEE dictionary term for "hit" (shown to be a synonym for cache hit). Here it shows that referencing data in cache eliminates the need to go to a secondary storage.

22. In regard to claim 20, Rotenberg discloses the method of claim 18 wherein the data source is system memory source. As shown above, a data cache exists in the system. The load commands have been shown above to be requests for data operands. It is inherent that a data cache first requests data from a data cache because its whole purpose of existence is to be a fast location for data retrieval. It is also inherent that if the data cache misses (the data is not in the cache), the data will be requested from the main system memory. Further proof is given in the included IEEE dictionary term for "hit" (shown to be a synonym for cache hit). Here it shows that referencing data in cache eliminates the need to go to a secondary storage, thus meaning that on a miss, the secondary storage is referenced. Since the disclosure of Rotenberg only discloses two data storage locations, the data cache and main memory, both shown above, the secondary storage is main system memory and thus is referenced.

23. In regard to claim 21, Rotenberg discloses the method of claim 17 further comprising: transmitting data to and from the load value queue using an error correction technique. The last two paragraphs of section 1.2 shows that the entire delay buffer (load value queue) concept is for checkpoint recovery (an error correction technique). Therefore, loads to and from the buffer use an error correction technique.

***Claim Rejections - 35 USC § 103***

24. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the

invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

25. Claims 8-12 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg in view of Tullsen (Simultaneous Multithreading).

26. In regard to claim 8,

a. Rotenberg discloses

- i. a pipelined, simultaneous and redundantly threaded ("SRT") processor, In the second column of section 1.1, the system is shown to run two explicit copies of the program run concurrently on the same processor resources. Further down it is shown that this is possible using simultaneous multithreading. Thus two copies of the same program (the term redundant is used on the top of the next page) are run in these simultaneous threads. Figure 5 shows that the system is broken up into pipeline stages and is thus pipelined. This figure, as well as figure 4, shows that the system includes a data cache or d-cache.
- ii. comprising a program counter configured to assign program count identifiers to instructions in each thread that are retrieved by the processor; Section 2.1.3 shows that a program counter is required for each thread, which inherently assign a program count identifier to instructions in the thread.
- iii. wherein said processor is configured to detect transient faults during program execution by executing instructions in at least two redundant copies of a program thread and wherein false errors caused by

incorrectly replicating fetched data in the redundant program threads are avoided by replicating the actual data retrieved from data fetch instructions in a first program thread for a second program thread. Section 1.2 shows that the disclosed system comprises multiple threads (streams) that run the same program at the same time with one thread having a slight lag and thus a set of instructions is run in a leading and trailing thread. The paragraph immediately under figure 3 shows that a delay buffer contains data and control information from the first run of the program (on the leading thread or A-stream). This is a clear differentiation of the result data and other control information such as that for predictions. In the paragraph directly above figure 3, it is shown that source operands (data) of instructions are predicted. In the paragraph directly below the figure again, it is stated that the delay buffer provides the R-stream (trailing thread) with perfect data flow prediction. Therefore, perfect data operand prediction must be provided, and thus the data operands themselves provided (replicated as the control information) in this delay buffer for the trailing thread.

- b. Rotenberg does not disclose
  - i. floating point execution units configured to execute floating point instructions;
  - ii. integer execution units configured to execute integer-based instructions;

- iii. load/store units configured to perform fetch and store operations to or from data sources such as a data cache, memory, and data registers;

However, Rotenberg does disclose multiple execution units (processing elements) as can be seen in figures 4 and 5.

- c. Tullen has taught a simultaneous multithreading system comprising
  - i. floating point execution units configured to execute floating point instructions;
  - ii. integer execution units configured to execute integer-based instructions;
  - iii. load/store units configured to perform fetch and store operations to or from data sources such as a data cache, memory, and data registers;

The last paragraph of page 393 shows that floating point, integer, and load/store units are included in the system for executing floating point, integer, and load/store instructions as given in table 1.

- d. Rotenberg has shown in section 2.1 that most of his design is derived from simultaneous multithreaded machines with the Tullen reference in particular (as seen in the bibliography to be reference 10) for techniques that are well established and understood and are seamlessly incorporated. This reference to Tullen for details of the design not disclosed by Rotenberg would have motivated one of ordinary skill in the art to modify the design of Rotenberg to include the details regarding the functional units disclosed by Tullen.

It would have been obvious to one of ordinary skill in the art to modify the design of Rotenberg to include the various functional units taught by Tullen because of the reference to this disclosure by Rotenberg concerning the details of his design.

27. In regard to claim 9, Rotenberg in view of Tullen discloses the SRT processor of claim 8 wherein the processor further comprises:

- a. a load value queue for storing the data values fetched in response to data fetch instructions in the first program thread; As shown above, the loaded operand data is stored in a delay buffer. The first paragraph of section 1.2 of Rotenberg shows that this buffer is in fact a queue and thus another appropriate name for the structure is a load value queue.
- b. wherein the load/store units place a duplicate copy of the data in the load value queue after fetching the data from the data source and wherein the load/store units access the load value queue and not the data source to fetch data values in response to data fetch instructions in the second program thread. Since all instructions store their data operands in the load value queue for correct operand prediction, load and store instructions also store data received in this queue and thus the load/store unit places the duplicate data in the queue. As shown above, the trailing thread fetches all data values from the load value queue (delay buffer) instead of the original data source. Therefore, the load and store instructions will look to this queue for data and thus the load/store unit accesses the queue.

28. In regard to claim 10, Rotenberg in view of Tullen discloses the SRT processor of claim 9 further comprising a register update unit; wherein the register update unit is configured to hold instructions in a queue until the instructions are executed and retired by the SRT processor and wherein the data fetched by the load/store units in response to data fetch instructions in the first program thread is not placed in the load value queue until the data fetch instructions in the first program thread retire from the register update unit. Section 2.1.1 of Rotenberg introduces a register-renaming scheme where multiple physical registers are mapped to the logical registers, by a register map, for the purpose of resolving data dependencies. Since the physical registers hold data (and thus are queues) to update the logical registers once all dependencies have been resolved, this structure is a register update unit. Figure 2 and the first paragraph of section 1.2 of Rotenberg show that the data stored in the load value queue (delay buffer) is pushed there when the results of the A-stream (leading thread) are committed or retired. Thus the data values must be stored in the register update unit until the instructions are retired and updates commence to both the registers and the load value queue.

29. In regard to claim 11,

- a. Rotenberg in view of Tullen discloses the SRT processor of claim 9 wherein data fetch instructions are executed in the same order in both the first and second program threads. Though section 1.3.1 details that instructions in the trailing thread (R-stream) may be executed highly in parallel due to perfect prediction, the instructions are not executed out-of-order, as can be seen in



figure 3. Therefore the trailing thread executes instructions including the data load commands in program order, as did the leading thread.

b. Rotenberg in view of Tullen as applied to claim 8 does not disclose wherein the SRT processor is an out-of-order processor capable of executing instructions in the most efficient order.

c. Tullen does disclose a simultaneous multithreaded processor that is an out-of-order processor capable of executing instructions in the most efficient order. Column 1, paragraph 2 on page 394 shows that out-of-order execution can be done and is supported and thus the processor is capable of out-of-order execution.

d. Rotenberg has shown in section 2.1 that most of his design is derived from simultaneous multithreaded machines with the Tullen reference in particular (as seen in the bibliography to be reference 10) for techniques that are well established and understood and are seamlessly incorporated. This reference to Tullen for details of the design not disclosed by Rotenberg would have motivated one of ordinary skill in the art to modify the design of Rotenberg to include the *capability* of out-of-order execution disclosed by Tullen.

It would have been obvious to one of ordinary skill in the art to modify the design of Rotenberg to include the capabilities of out-of-order execution by Tullen because of the reference to this disclosure by Rotenberg concerning the details of his design.

30. In regard to claim 12, Rotenberg in view of Tullen discloses the SRT processor of claim 11 wherein the load value queue is a FIFO buffer and data is transmitted to and

from the buffer using an error correction technique. The first paragraph of section 1.2 show that the load value queue is a FIFO delay buffer. The next two paragraphs of section 1.2 shows that the entire delay buffer (load value queue) concept is for checkpoint recovery (an error correction technique). Therefore, loads to and from the buffer use an error correction technique.

31. In regard to claim 14, Rotenberg in view of Tullen discloses the SRT processor of claim 12 wherein if the load value queue becomes full, the first thread is stalled to prevent more data values from entering the load value queue; and wherein if the load value queue becomes empty, the second thread is stalled to allow data values to enter the load value queue. Near the end of the summary section (section 4), it is disclosed that if the delay buffer (load value queue) is full, the R-stream (trailing thread) is given priority to access the fetch/dispatch pipeline, meaning that the A-stream or leading thread is temporarily halted in favor of the trailing thread so that it can issue and dispatch instructions. It is also disclosed that if the delay buffer (load value queue) is not full (where empty is an instance of this), the A-stream (leading thread) is given priority retire a trace, meaning that the R-stream or trailing thread is temporarily halted in favor of the leading thread when it can retire a trace of instructions.

32. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg in view of Tullen as applied to claims 11-12 above, and further in view of Merchant (6,665,792).

33. In regard to claim 13,

- a. Rotenberg in view of Tullen discloses the SRT processor of claim 12 wherein the individual load value entries in the load value queue comprise: the data source from which the data was fetched; Since, as shown above, the trailing thread executes the same instructions as the leading thread,
  - i. an address indicating the physical location in the data source from which the data was fetched; an address indicating the physical location in the trailing thread inherently locates the correct data with the same address. Thus the address must be stored in the buffer.
  - ii. and the data value that was retrieved by the load/store units when the fetch command was executed (as shown above in regards to the independent claim).
- b. Rotenberg in view of Tullen does not disclose that the size of the data value is stored in the buffer;
- c. Merchant has taught in column 6, lines 10-13 show that a load buffer stores the size of the data.
- d. The system of Rotenberg in view of Tullen, as shown above, stores operand data in the buffers. In order to handle the varying sizes of operand data for execution, a field indicating the size of the operand would speed up recognition on how to handle the data. This ability to quickly recognize operand size would have motivate one of ordinary skill in the art to modify the design of Rotenberg in view of Tullen to include the data size in the load buffer as taught by Merchant.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Rotenberg in view of Tullen to store the data size in the load buffer so that operand sizes are quickly recognized and the operands handled correctly.

34. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rotenberg in view of Merchant (6,665,792).

35. In regard to claim 18,

- a. Rotenberg discloses the method of claim 17 further comprising:
  - iii. storing the data value in a FIFO buffer; The first paragraph of section 1.2 show that the load value queue is a FIFO delay buffer.
  - iv. and storing the following information with each entry in the buffer:
    - (1) an address indicating the physical location in the data source from which the data was fetched; Since, as shown above, the trailing thread executes the same instructions as the leading thread, the trailing thread inherently locates the correct data with the same address. Thus the address must be stored in the buffer.
    - (2) and the data value that was retrieved by the data request in the leading thread (as shown above in regards to the independent claim).
- b. Rotenberg does not disclose that the size of the data value is stored in the buffer;
- c. Merchant has taught in column 6, lines 10-13 show that a load buffer stores the size of the data.

d. The system of Rotenberg, as shown above, stores operand data in the buffers. In order to handle the varying sizes of operand data for execution, a field indicating the size of the operand would speed up recognition on how to handle the data. This ability to quickly recognize operand size would have motivate one of ordinary skill in the art to modify the design of Rotenberg to include the data size in the load buffer as taught by Merchant.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Rotenberg to store the data size in the load buffer so that operand sizes are quickly recognized and the operands handled correctly.

### ***Conclusion***

36. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

37. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. The following patent has been cited to further show the art with respect to simultaneous redundant multithreading processors.

US Pat No 5,193,175 to Cutts teaches a simultaneous multithreading pipeline that runs the same instruction stream on each thread (or pipeline) where each thread is staggered from the others.

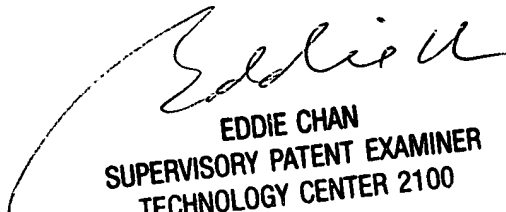
Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (703)305-7305. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703)305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl  
Examiner  
Art Unit 2183

SFG  
March 4, 2004

  
EDDIE CHAN  
SUPERVISORY PATENT EXAMINER  
TECHNOLOGY CENTER 2100